The University of Alabama in Huntsville
Electrical and Computer Engineering Department
CPE 221 01
Sample Test 1 Solution

This test is closed book, closed notes. You may not use a calculator. You should have the 6 page ARM Instruction Reference. You must show your work to receive full credit.

1. (1 point) An _**embedded**_ computer is one that is dedicated to doing one task, such as the one found in washing machines.
2. (1 point) _**Registers**_ are used to hold data when faster access than main memory is needed.
3. (1 point) _**Branch**_ instructions can alter the normal flow of control from executing the next instruction in sequence.
4. (1 point) In RTL, the symbol _**←**_ is used to indicate a data transfer.
5. (1 point) _**Indirect, Indexed, PC-relative, direct, immediate**_ is an example of an addressing mode found in processors.
6. (10 points) Represent +193 and -259 as signed (2s complement) 16-bit numbers

```
        193                          259
 /2      96   1              /2      129   1
 /2      48   0              /2       64   1
 /2      24   0              /2       32   0
 /2      12   0              /2       16   0
 /2       6   0              /2        8   0
 /2       3   0              /2        4   0
 /2       1   1              /2        2   0
 /2       0   1              /2        1   0
                             /2        0   1
```

```
+193 = 0000 0000 1100 0001 = 0x00C1
+259 = 0000 0001 0000 0011
-259 = 1111 1110 1111 1101 = 0xFEFD
```

7. (10 points) If `r1 = 0x00FF` and `r2` = 4, what is the value of `r0` after each of the following instructions has been executed (assume that each instruction uses the same data)?

(a) `ADD r0, r1, r1, LSL #2`

`0x0000 00FF + 0x0000 03FC = 0x0000 04FB`

(b) `ADD r0, r1, r1, ROR #17`

`0x0000 00Ff + 0x007F 1000 = 0x007F 80FF`

(c) `ADD r0, r1, r1, LSR r2`

`0x0000 00FF + 0x0000 000F = 0x0000 010E`

8. (10 points) For each of the following operations on 6 bit signed numbers, calculate the values of the C, Z, V, and N flags

(a) 001011 + 001101

```
 001111
  001011
+ 001101
  011000
```

VCZN = 0000

(b) 111111 + 000001

```
 111111
  111111
+ 000001
  000000
```

VCZN = 0110

9. (15 points) Assume that `r2` contains the initial value `0xFF001000`. Explain the effect of each of the following instructions, and give the value in `r2` after each instruction executes. Use register transfer notation.

(a) `LDR r1, [r2]`

r1 ← M[r2], r2 = 0xFF00 1000

(b) `STR r1, [r2, #2_10010]`

M[r2 + 0x12] ← r1, r2 = 0xFF00 1000

(c) `STR r1, [r2, #0x24]!`

M[r2 + 0x24] ← r1, r2 = 0xFF00 1024

(d) `STR r1, [r2], #8`

M[r2] ← r1 = 0xFF00 1008

(e) `STR r1, [r2, r0, ASR #8]`

M[r2 + r0>>8] ← r1, r2 = 0xFF00 1000

10. (25 points) Consider the following ARM program. Trace the values of the registers shown as they change during program execution. Also, trace the writes to memory by the `STR` instruction. There may be unused columns or rows in the tables. If you need to add columns or rows, you may do so. `DCD 1` reserves one word of storage and sets it equal to 1. `SPACE 3` reserves 3 bytes of memory but does not give those bytes a value.

```
        AREA    PROB_10, CODE, READONLY
        ADR     r0, x
        ADR     r1, y
        ADR     r2, z
        LDR     r3, size
        LDR     r4, i
loop    SUBS    r5, r4, r3
        BPL     done
        LDR     r5, [r0]
        LDR     r6, [r1]
        ADD     r5, r5, r6
        STR     r5, [r2]
        ADD     r0, r0, #4
        ADD     r1, r1, #4
        ADD     r2, r2, #4
        ADD     r4, r4, #1
        B       loop
done    B       done
size    DCD     6
i       DCD     0
x       DCD     100, 3, -1, 2, 4, 4
y       DCD     -53, 247, 95, -7, 481, 91
z       SPACE   24
        END
```

r0	76			80			84			88			92			96			100
r1	100			104			108			112			116			120			124
r2	124			128			132			136			140			144			148
r3	6																		
r4	0			1			2			3			4			5			6
r5	-6	100	47	-5	3	250	-4	-1	94	-3	2	-5	-2	4	485	-1	4	95	0
r6		-53			247			95			-7			481			91		

Results of the STR instruction.

Memory Address	Contents
124	47
128	250
132	94
136	-5
140	485
144	95

11. (25 points) Complete the ARM assembly language program below so that it implements the following C++ statements.

```
//
//      This program examines two arrays, element by element and copies the
//      largest number of each pair into a third array.
//
const int size = 10;
int x[size] = {100, 3, -1, 2, 4, 4, 2, -1, 3, 100};
int y[size] = {-53, 247, 95, -7, 481, 91, -33, 1500, 29, -83};
int z[size];
int i;

for (i = 0; i < size; i++)
  if (x[i] > y[i])
    z[i] = x[i];
  else
    z[i] = y[i];
```

```
        AREA    PROB_11, CODE, READONLY
        ENTRY
        ADR     r0, x
        ADR     r1, y
        ADR     r2, z
        LDR     r3, size
        LDR     r4, i
loop    CMP     r4, r3
        BPL     done
        LDR     r5, [r0], #4
        LDR     r6, [r1], #4
        CMP     r5, r6
        STRGT   r5, [r2], #4
        STRLE   r6, [r2], #4
        ADD     r4, r4, #1
        B       loop
done    B       done
x       DCD     100, 3, -1, 2, 4, 4, 2, -1, 3, 100
y       DCD     -53, 247, 95, -7, 481, 91, -33, 1500, 29, -83
z       SPACE   40
i       DCD     0
size    DCD     10
```